

APPENDIX A

ALGORITHM DESCRIPTION

MULTIPROD performs multiple matrix multiplications. This document describes how they are performed, often by using the TIMES operator (.*) rather than MTIMES (*). Indeed, TIMES works with N-D arrays, while MTIMES works only with 2-D arrays. Also, the handle @TIMES is accepted by BSXFUN (a builtin function performing multiple operations with singleton expansion enabled), while @MTIMES is not. These examples do not require array expansion (AX), except for the case solved using the “2-D squashing” technique.

In the following tables, the sequence of steps “SX + TIMES” is executed using the builtin function BSXFUN:

$C = \text{BSXFUN}(@\text{TIMES}, A, B)$
























LEGENDA

ADDSINGLETON	Insertion of singleton dimension by reshaping.
SWAPPING	Swapping of internal dimensions by reshaping (permutation not needed).
SX (singleton expansion)	Both arrays are replicated along their singl. dimensions to match each other's size.
TIMES	Element-by-element multiplication.
SUM	Sum along dimension D1 or D2.
DELSINGLETON	Removal of singleton dimension by reshaping.














EXCEPTIONS

TIMES	If A or B or both are scalars (i.e. they contain only one element), $C = A .* B$ is executed.
MTIMES	If A and B are both 2-D, and their internal dimensions are [1 2], $C = A * B$ is executed.














2-D BLOCKS BY 2-D BLOCKS

		Case 1	Case 2	Case 3	Case 4 (inner)	Case 5 (outer)	Case 6	Case 7
INPUT	A	$6 \times (3 \times 4)$	$6 \times (1 \times 4)$	$6 \times (3 \times 4)$	$6 \times (1 \times 4)$	$6 \times (3 \times 1)$	$6 \times (1 \times 1)$	$6 \times (1 \times 1)$
	B	$6 \times (4 \times 2)$	$6 \times (4 \times 2)$	$6 \times (4 \times 1)$	$6 \times (4 \times 1)$	$6 \times (1 \times 2)$	$6 \times (R \times S)$	$6 \times (1 \times 1)$
ADDSINGL. or SWAPPING								
	A	$6 \times (3 \times 4 \times 1)$						
	B	$6 \times (1 \times 4 \times 2)$						
								
SX	A	$6 \times (3 \times 4 \times 2)$	$6 \times (4 \times 2)$	$6 \times (3 \times 4)$		$6 \times (3 \times 2)$	$6 \times (R \times S)$	$6 \times (P \times Q)$
	B	$6 \times (3 \times 4 \times 2)$	$6 \times (4 \times 2)$	$6 \times (3 \times 4)$		$6 \times (3 \times 2)$	$6 \times (R \times S)$	$6 \times (P \times Q)$
TIMES								
	C	$6 \times (3 \times 4 \times 2)$	$6 \times (4 \times 2)$	$6 \times (3 \times 4)$	$6 \times (4 \times 1)$	$6 \times (3 \times 2)$	$6 \times (R \times S)$	$6 \times (P \times Q)$
SUM								
	C	$6 \times (3 \times 1 \times 2)$	$6 \times (1 \times 2)$	$6 \times (3 \times 1)$	$6 \times (1 \times 1)$			
DELSINGL.								
	C	$6 \times (3 \times 2)$						

2-D BLOCKS BY 1-D BLOCKS

		Case 3b	Case 4b (inner)	Case 6b		Case 7b
INPUT	A	$6 \times (3 \times 4)$	$6 \times (1 \times 4)$	$6 \times (1 \times 1)$	$6 \times (P \times Q)$	$6 \times (1 \times 1)$
	B	$6 \times (4)$	$6 \times (4)$	$6 \times (4)$	$6 \times (1)$	$6 \times (1)$
						
ADDSINGL. or DELSINGL.	A	$6 \times (3 \times 4)$			$6 \times (P \times Q)$	$6 \times (1 \times 1)$
	B	$6 \times (1 \times 4)$			$6 \times (1 \times 1)$	$6 \times (1 \times 1)$
			$6 \times (4)$ $6 \times (4)$	$6 \times (1)$ $6 \times (4)$		
SX	A	$6 \times (3 \times 4)$		$6 \times (4)$	$6 \times (P \times Q)$	
	B	$6 \times (3 \times 4)$		$6 \times (4)$	$6 \times (P \times Q)$	
						
TIMES	C	$6 \times (3 \times 4)$	$6 \times (4)$	$6 \times (4)$	$6 \times (P \times Q)$	$6 \times (1 \times 1)$
						
SUM	C	$6 \times (3 \times 1)$	$6 \times (1)$			
						
DELSINGL.	C	$6 \times (3)$				

1-D BLOCKS BY 2-D BLOCKS

		Case 2c	Case 4c (inner)	Case 6c		Case 7c
INPUT	A	$6 \times (4)$	$6 \times (4)$	$6 \times (4)$	$6 \times (1)$	$6 \times (1)$
	B	$6 \times (4 \times 2)$	$6 \times (4 \times 1)$	$6 \times (1 \times 1)$	$6 \times (R \times S)$	$6 \times (1 \times 1)$
						
ADDSINGL. or DELSINGL.	A	$6 \times (4 \times 1)$			$6 \times (1 \times 1)$	$6 \times (1 \times 1)$
	B	$6 \times (4 \times 2)$			$6 \times (R \times S)$	$6 \times (1 \times 1)$
			$6 \times (4)$ $6 \times (4)$	$6 \times (4)$ $6 \times (1)$		
SX	A	$6 \times (4 \times 2)$		$6 \times (4)$	$6 \times (R \times S)$	
	B	$6 \times (4 \times 2)$		$6 \times (4)$	$6 \times (R \times S)$	
						
TIMES	C	$6 \times (4 \times 2)$	$6 \times (4)$	$6 \times (4)$	$6 \times (R \times S)$	$6 \times (1 \times 1)$
						
SUM	C	$6 \times (1 \times 2)$	$6 \times (1)$			
						
DELSINGL.	C	$6 \times (2)$				

1-D BLOCKS BY 1-D BLOCKS

		Case 4d (inner)	Case 5d (outer)	Case 6d		Case 7d
INPUT	A	6×(4)	6×(3)	6×(1)	6×(3)	6×(1)
	B	6×(4)	6×(2)	6×(3)	6×(1)	6×(1)
ADDSINGL.	A		6×(3×1)	↓	↓	
	B		6×(1×2)			
SX	A	↓	6×(3×2)	6×(3)	6×(3)	↓
	B		6×(3×2)	6×(3)	6×(3)	
TIMES	C	6×(4)	6×(3×2)	6×(3)		6×(1)
		↓				
SUM	C	6×(1)				
DELSINGL.	C					

2-D SQUASHING

The "2-D squashing" technique is used if one and only one of the arrays (A or B) is single-block (e.g. a (4×3), 1×(4×3), or 1×(4) array), and not a scalar. This is the simplest and computationally most efficient form of virtual AX. The 2-D squashing algorithm is a generalization of a more specific algorithm designed to deal with the fixed array sizes shown below. This specific algorithm, whose author was allegedly a contributor of a MATLAB newsgroup (Acklam, 2003), was suggested to me by Jinhui Bai.

	Array size	Jinhui Bai's implementation
A	2-D (P×Q)	Ctemp = A * B(:, :);
B	3-D (Q×S)×N	C = RESHAPE(Ctemp, SIZE(A,1), SIZE(B,2), []);
A	3-D (P×Q)×N	Atemp = PERMUTE(A, [1 3 2]);
B	2-D (Q×S)	Atemp = RESHAPE(Atemp, [], SIZE(A,2));
		Ctemp = Atemp * B;
		Ctemp = RESHAPE(Ctemp, SIZE(A,1), SIZE(A,3), SIZE(B,2));
		C = IPERMUTE(Ctemp, [1 3 2]);

This code assumes that block size is 2-D and the internal dimensions of both A and B are [1 2]. In MULTIPROD, neither A nor B can be assumed to meet these conditions. MULTIPROD removes leading singletons from single-block arrays with higher IDs, and expands 1-D blocks to 2-D by adding a singleton dimension. Then, it calls the **squash2D_mtimes** subfunction, which performs the following operations:

- the multi-block array is rearranged from N-D to 2-D (by permutation and reshaping),
- MTIMES is applied,
- the result is rearranged back to N-D (by reshaping and permutation).

Squash2D_mtimes accepts N-D multi-block arrays (N>2), and uses optimal permutation order ("partial shifting"), to minimize processing time (Appendix B). Its output always contains 2-D blocks. When output containing 1-D blocks is expected, MULTIPROD removes a singleton dimension to obtain it (see below).

EITHER A OR B IS A SINGLE-BLOCK ARRAY

		Case 1	Case 2	Case 3	Case 4 (inner)	Case 5 (outer)
INPUT	A	$3 \times (P \times Q) \times 2$	$3 \times (Q) \times 2$	$3 \times (1) \times 2$	$3 \times (P \times Q) \times 2$	$3 \times (P) \times 2$
	B	$1 \times 1 \times (R \times S)$	$1 \times 1 \times (R \times S)$	$1 \times 1 \times (R \times S)$	$1 \times 1 \times (R)$	$1 \times 1 \times (Q)$
		↓	↓	↓	↓	↓
DELSINGL.	A	$3 \times (P \times Q) \times 2$	$3 \times (Q) \times 2$	$3 \times (1) \times 2$	$3 \times (P \times Q) \times 2$	$3 \times (P) \times 2$
	B	$(R \times S)$	$(R \times S)$	$(R \times S)$	(R)	(Q)
ADDSINGL.	A		$3 \times (1 \times Q) \times 2$	$3 \times (1 \times 1) \times 2$	$3 \times (P \times Q) \times 2$	$3 \times (1 \times P) \times 2$
	B		$(R \times S)$	$(R \times S)$	$(R \times 1)$	$(P \times 1)$
		↓	↓	↓	↓	↓
SQUASH2D_ MTIMES	C	$3 \times (P \times S) \times 2$	$3 \times (1 \times S) \times 2$	$3 \times (R \times S) \times 2$	$3 \times (P \times 1) \times 2$	$3 \times (1 \times 1) \times 2$
			↓		↓	↓
DELSINGL.	C		$3 \times (S) \times 2$		$3 \times (P) \times 2$	$3 \times (1) \times 2$

BOTH A AND B ARE SINGLE-BLOCK ARRAYS (RARE CASE)

		Case 1b	Case 2b	Case 10	Case 11 (inner)	Case 12 (outer)
INPUT	A	$1 \times 1 \times (P \times Q)$	$1 \times 1 \times (Q)$	$1 \times 1 \times (P \times Q)$	$1 \times 1 \times (P)$	$1 \times 1 \times (P)$
	B	$1 \times 1 \times (R \times S)$	$1 \times 1 \times (R \times S)$	$1 \times 1 \times (R)$	$1 \times 1 \times (P)$	$1 \times 1 \times (Q)$
		↓	↓	↓	↓	↓
DELSINGL.	A	$(P \times Q)$	(Q)	$(P \times Q)$	(P)	(P)
	B	$(R \times S)$	$(R \times S)$	(R)	(P)	(Q)
ADDSINGL.	A		$(1 \times Q)$	$(P \times Q)$	$(1 \times P)$	$(P \times 1)$
	B		$(R \times S)$	$(R \times 1)$	$(P \times 1)$	$(1 \times Q)$
		↓	↓	↓	↓	↓
MTIMES	C	$(P \times S)$	$(1 \times S)$	$(P \times 1)$	(1×1)	$(P \times Q)$
		↓	↓	↓	↓	↓
ADDSINGL.	C	$1 \times 1 \times (P \times S)$	$1 \times 1 \times (1 \times S)$	$1 \times 1 \times (P \times 1)$	$1 \times 1 \times (1 \times 1)$	$1 \times 1 \times (P \times Q)$
DELSINGL.	C		$1 \times 1 \times (S)$	$1 \times 1 \times (P)$	$1 \times 1 \times (1)$	

REFERENCE

P. J. Acklam (2003). MATLAB array manipulation tips and tricks. Web site:
<http://home.online.no/~pjacklam/matlab/doc/mtt/index.html>.
Retrieved on 2009 Feb 13.